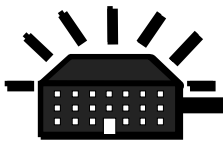
ORACLE®

Bitmap Indexing in Oracle Data Warehousing

Håkan Jakobsson
SQL-DSS Group
Server Technologies Division

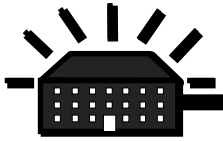


Overview of Presentation

- Data warehousing and OLTP in one engine.
- What is bitmap indexing?
- Technical implementation issues.
- Parallel bitmap star joins.
- Conclusion.

ORACLE®

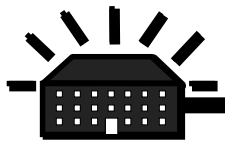
Integrated vs. Specialized Products for Warehousing



- Should/can bitmap indexes (and other DSS features) be integrated into a general purpose engine (like Oracle)?
- A specialized RDBMS-product is (supposedly) better at something.
- But likely to be weak at fundamental RDBMS features, e.g., security, procedural language support, backup/recovery, support for updates, general query processing, support for large number of users, etc.
- Maintaining multiple code lines is highly undesirable.

ORACLE®

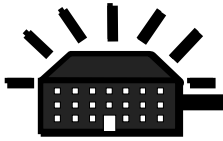
Integrated vs. Specialized Products for Warehousing



- Moreover, customers familiar with Oracle like to take advantage of their in-house expertise.
- A product that can't do both OLTP and DSS is unsuitable foundation for an Operational Data Store.
- Bottom line: Those vendors that can tend to go for an integrated product.

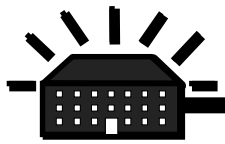
ORACLE®

Why Update Capability Is Useful



- Data in warehouse is modified to reflect changes in (our knowledge about) the world.
- Preliminary data, e.g., sales figures.
- Data subject to amendment, e.g., tax returns, airline reservations.
- Sales reorg: Change organization of sales districts and make historical data reflect new organization.

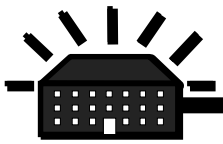
ORACLE®



What Is a Bitmap Index?

- Old concept. In several commercial products.
- Key idea: Index entries contain bitmap (or bit vector) of table instead of list of rowids.
- Each bit in bitmap corresponds to a row in table.
- RID-list: <CA> <$rowid_1$><$rowid_6$><$rowid_{15}$>...
- Bitmap: <CA> <1000010000000001…>
- Compress sparse bitmaps (long sequences of zeros).

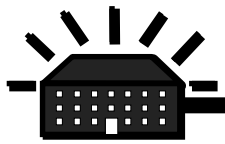
ORACLE®



Basic Observations

- Bitmap representation can save a lot of space over rowid lists, especially for low cardinality data.
- Lends itself to fast boolean operations for combining bitmap from different index entries.
- Requires a mapping function between physical rows and bits in bitmap.

ORACLE®

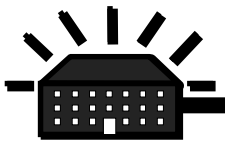


Bitmap Indexes in Oracle

- Fully updateable.
- Support for parallelism, both for queries and creation.
- Any cardinality data.
- Can be combined with regular B-tree indexes in same access path.

ORACLE®

Oracle's Bitmap Access Methods

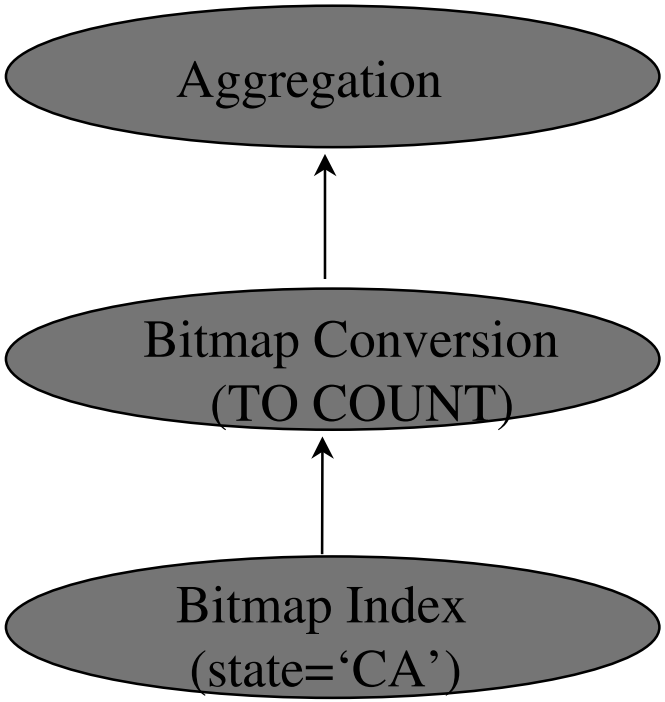


- Bitmap Index Probe
- Bitmap AND
- Bitmap OR
- Bitmap Minus
- Bitmap Merge
- Bitmap Conversion

ORACLE®

Equality Predicates

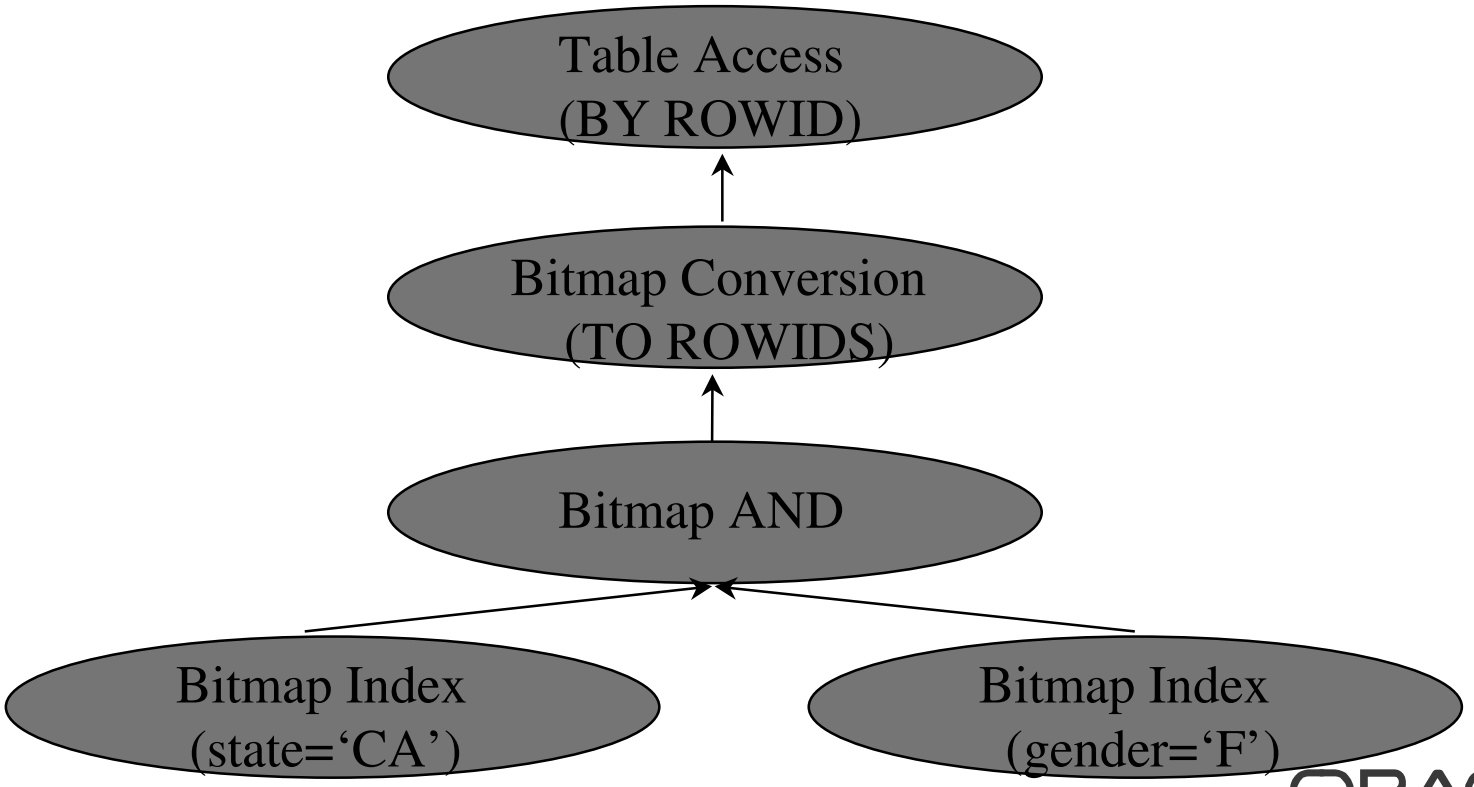
select count(*)
from Customer
where State = ‘CA’



ORACLE®

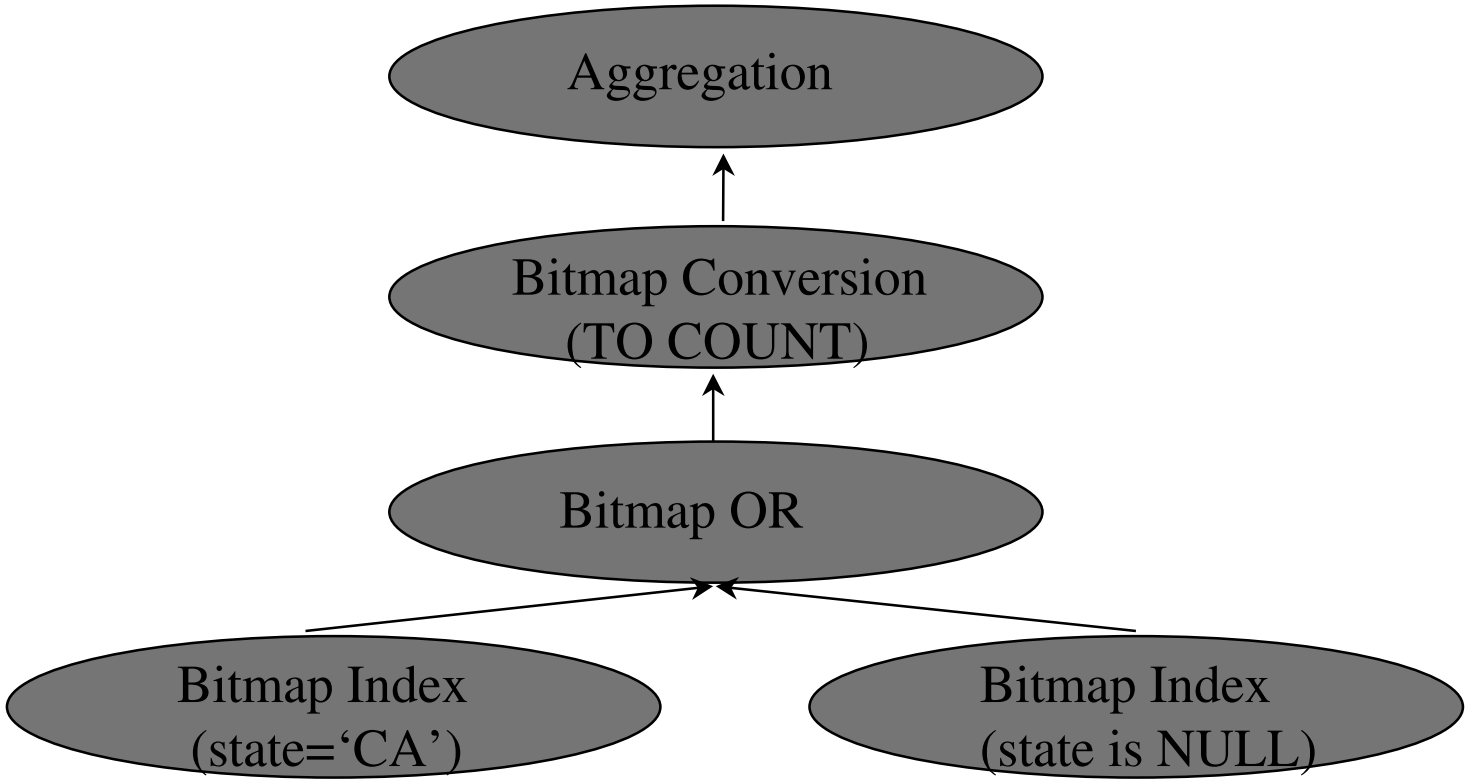
AND Predicates

select Name from Customer
where state = ‘CA’ and gender = ‘F’



OR Predicates

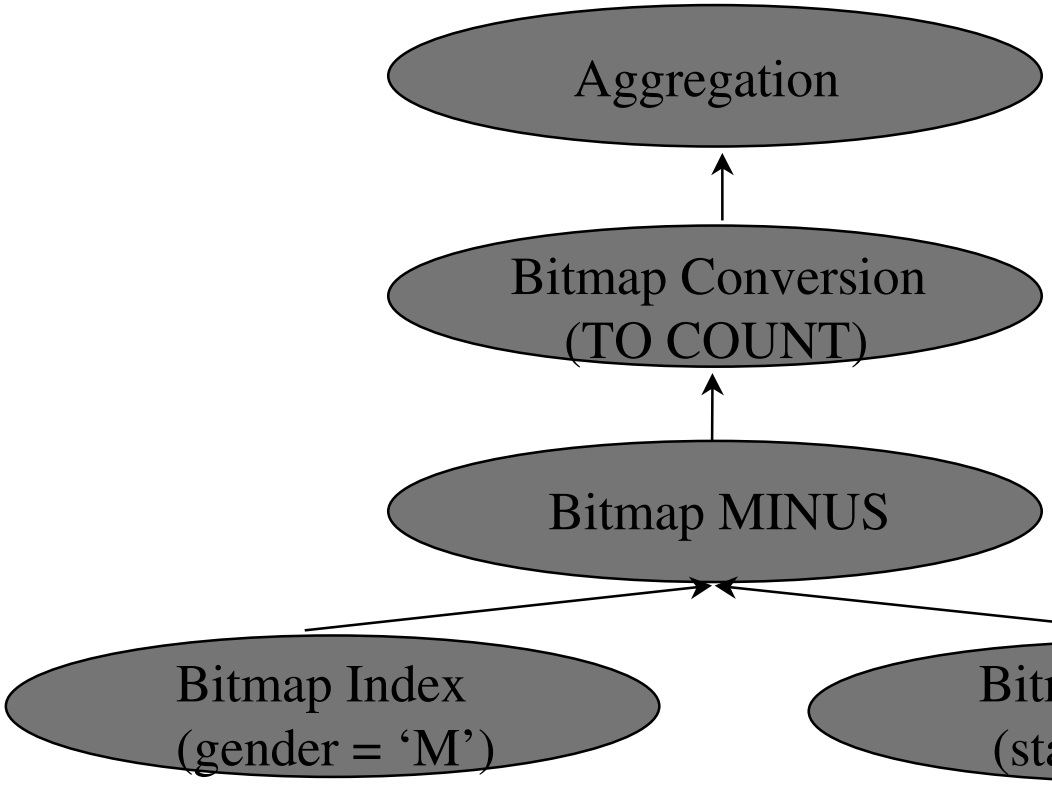
select count(*) from Customer
where state = ‘CA’ or state is NULL



ORACLE®

NOT EQUAL Predicates

select count(*) from Customer
where gender = 'M' and state != 'CA'

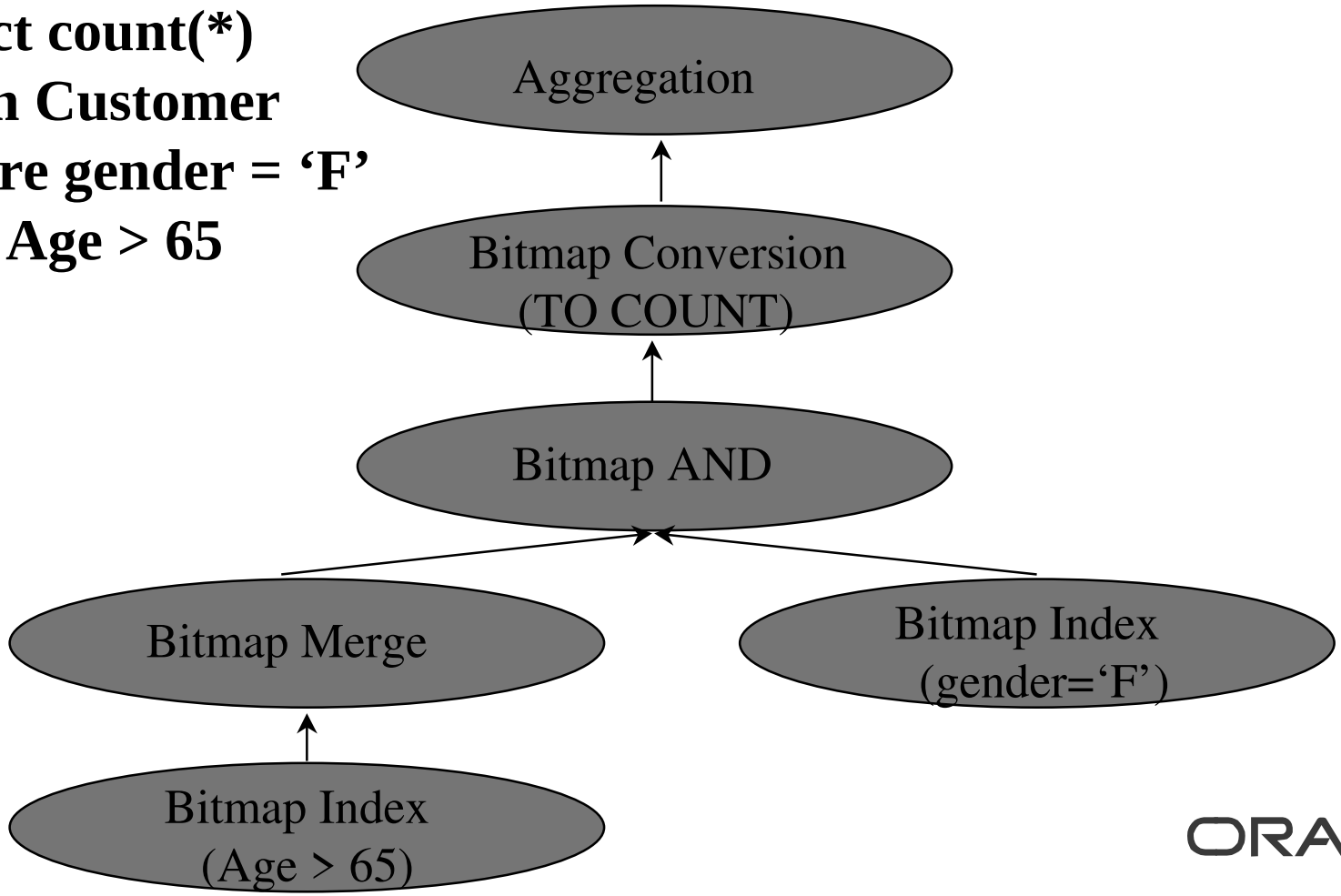


Assuming state is declared NOT NULL.

ORACLE®

Range Predicates

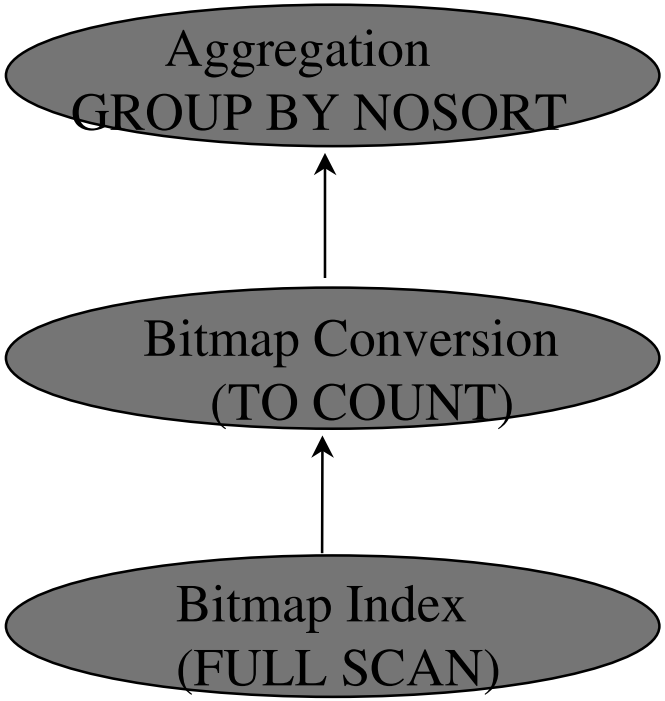
select count(*)
from Customer
where gender = 'F'
and Age > 65



ORACLE®

GROUP BY Queries

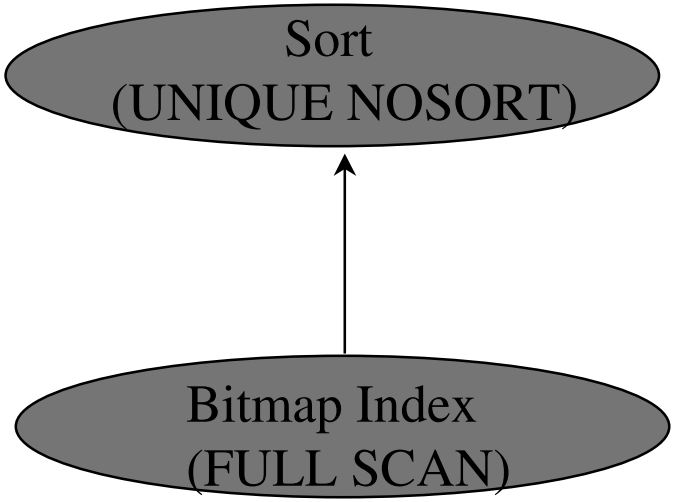
select state, count(*)
from Customer
group by State



ORACLE®

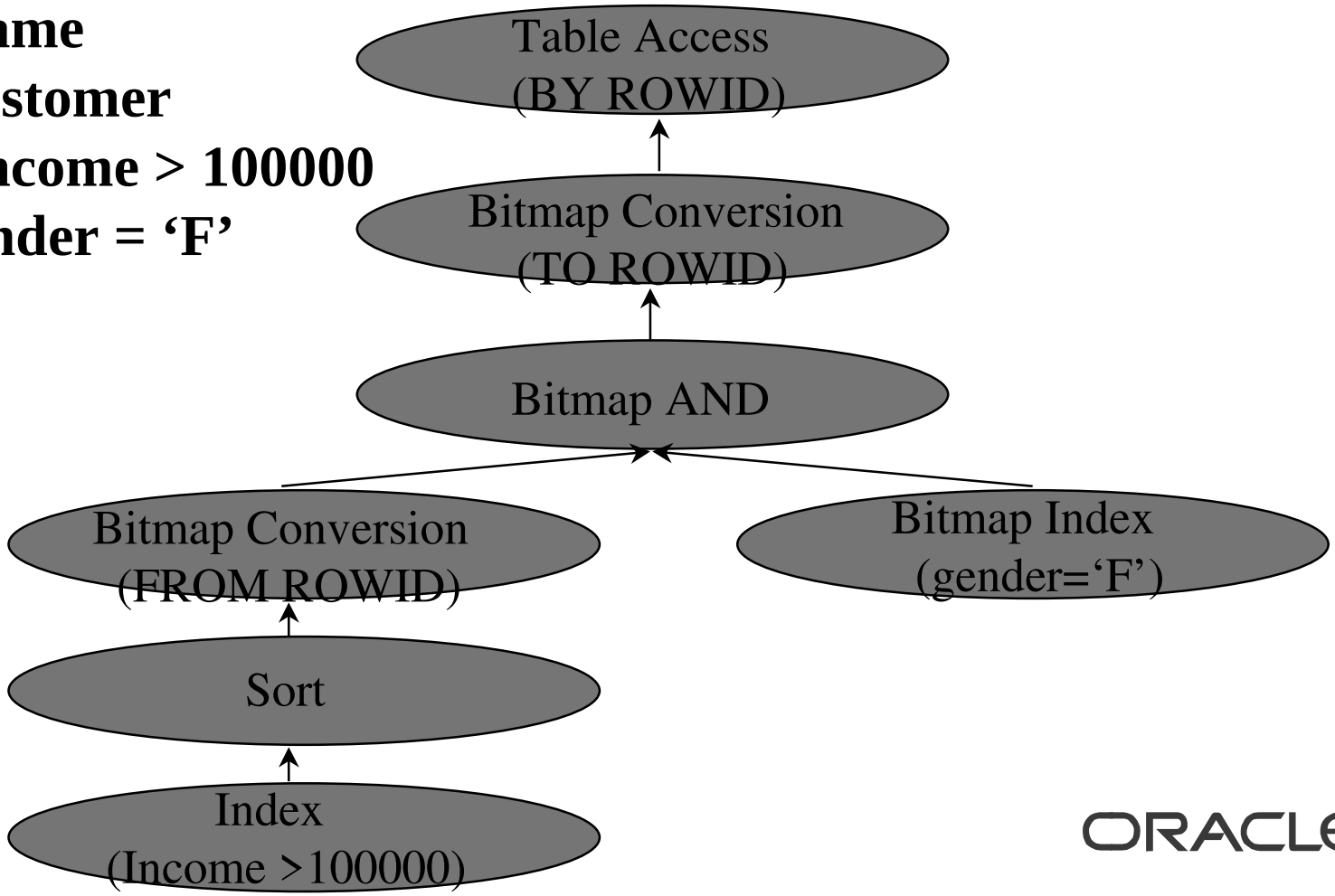
Distinct Queries

select distinct State
from Customer



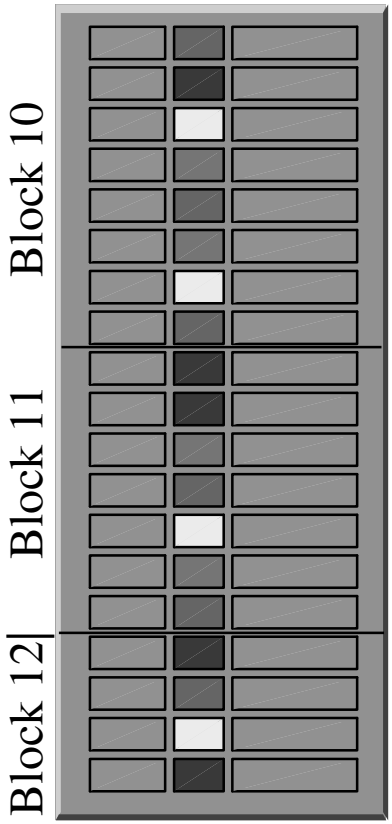
Combining Predicates

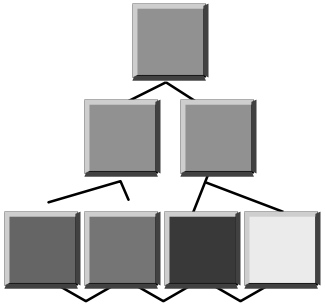
select Name
from Customer
where Income > 100000
and Gender = 'F'



ORACLE®

Bitmap Indexes On Disk





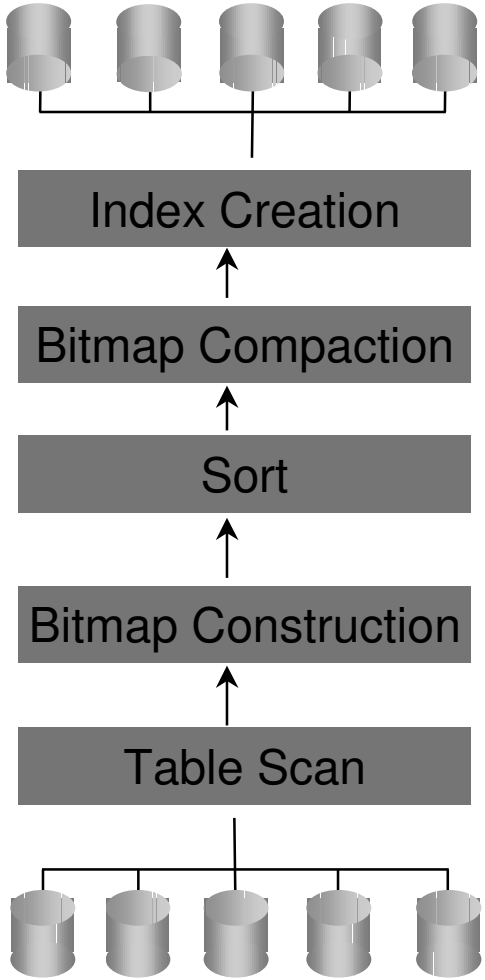
Bitmaps stored in leaves of b-tree indexes as bitmap segments.

key	start rowid	end rowid	bitmap
<Blue,	10.0,	12.8,	10001001000100101 00>
<Green,	10.0,	12.8,	00010100001001000 00>
<Red,	10.0,	12.8,	01000000110000010 01>
<Yellow,	10.0,	12.8,	00100010000010000 10>

All bitmaps compressed using patented algorithm.

ORACLE®

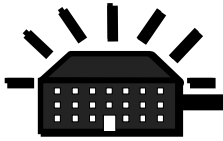
Bitmap Index Creation



Fully Parallelized:
- **by rowid ranges for non-partitioned tables**
- **by partition for partitioned tables**

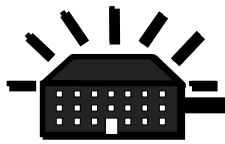
ORACLE®

Updates and Mapping Rows to Bitmaps



- If new rows are inserted, the relationship between bits in bitmap and rows in table could be destroyed.
- Solution: Don’t map bits to existing rows, but to potential rows based on the maximum number of rows per block.
- Compression will take care of excessive zeros.

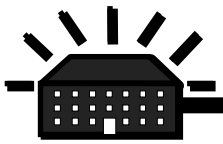
ORACLE®



Updates to Bitmap Indexes

- Batched updates -- each bitmap only modified once per statement. Cf. insert into table (select ...)
- Normal Oracle consistency model, but bitmap segment locking granularity. Hence, not very suitable for high concurrency OLTP applications.
- But, no read locks in Oracle, so readers don't block writers. So, locking granularity only an issue for updates.
- May be faster or slower than updates to regular B-tree indexes depending on effect of batching.

ORACLE®



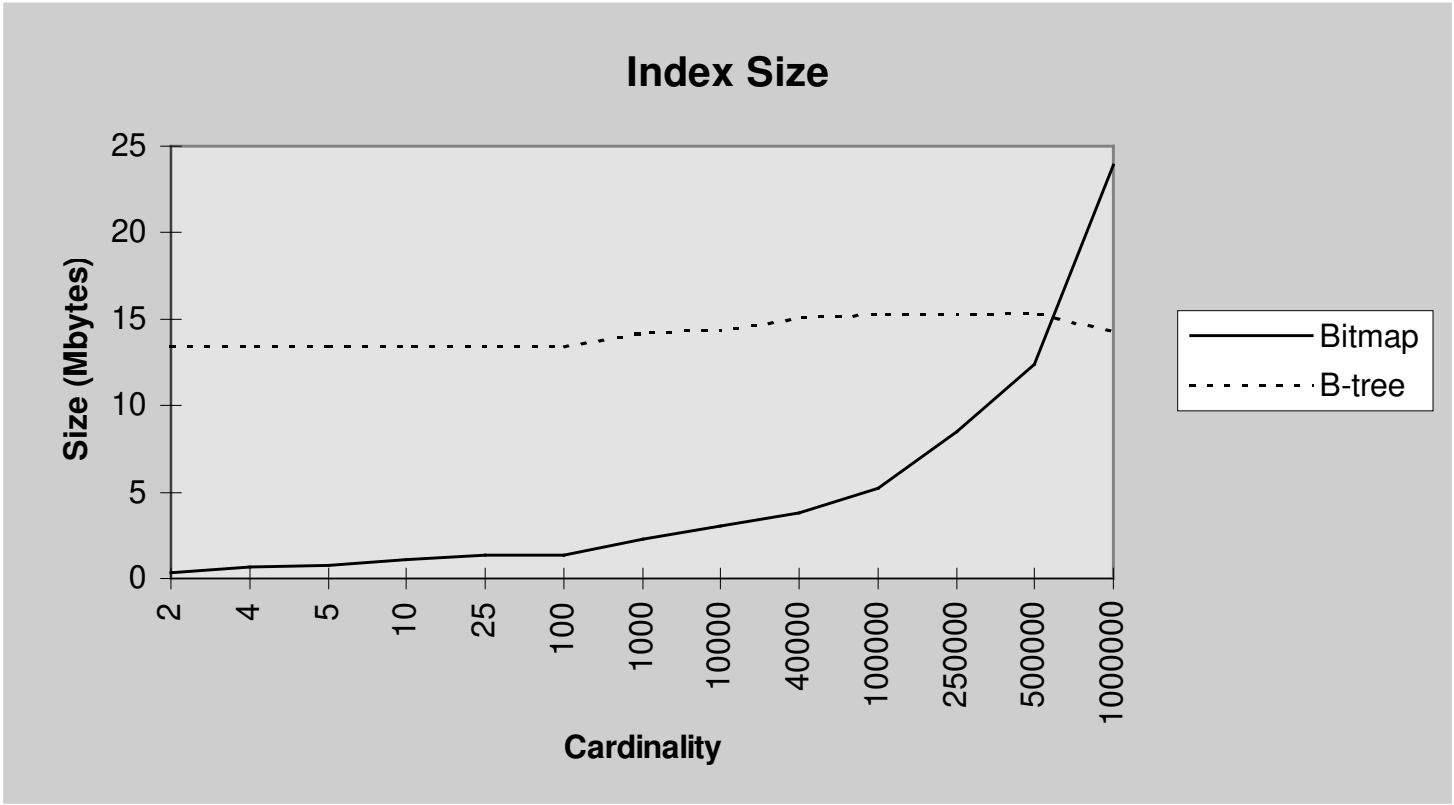
Compression

- Patented algorithm due to Gennady Antoshenkov.
- Store verbatim bitmap if distance to next 1 is small. Store length gap if distance is large.
- Adjusting representation in the compression algorithm means that same type of bitmap index can support any cardinality data efficiently.
- Clever encoding means very low overhead.
- Lends itself to efficient boolean operations.

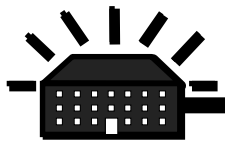
ORACLE®

Index Size

1 Million Rows



ORACLE®

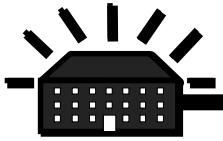


Star Join Queries

- New for Oracle 8.0.
- A query transformation technique to process multi-table joins in star schemas.
- Use bitmap indexes on fact table join columns.
- The transformation is completely cost based and the optimizer also considers alternative access path, such as a full table scan of the fact table.

ORACLE®

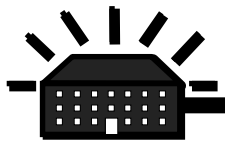
Star Join Queries (cont'd)



Basic Idea

- Use bitmap indexes as access path for fact table.
- Generate new subqueries as sources of keys to drive the bitmap index accesses.
- OR the bitmaps for each dimension. Then AND the ORed bitmaps.
- The new subqueries may obviate the need for some of the joins (join-back elimination).
- Reject plan with transformation if other plan is better.
- Cf. [O'Neil & Graefe, 1995]

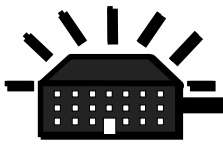
ORACLE®



Star Join Queries (cont'd)

```
select  ...
from fact_table
where (fkey1 = 2344 or fkey1 = 5454 or fkey1 = 5457 or …)
    and (fkey2 = 45545 or fkey2 = 78745 or fkey2 = 79800 or …)
    and (fkey3 = 1355 or fkey3 = 12378 or fkey3 = 31255 or …)
```

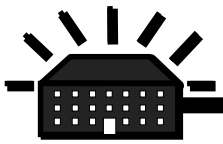
ORACLE®



Star Join Queries (cont'd)

```
select c.Name, s.Price
from Employee e, Product p, Customer c, Sales s
where e.EmpId = s.EmpId
    and p.ProductId = s.ProductId
    and c.CustomerId = s.CustomerId
    and c.Income > 100000
    and p.Supplier = 'Oracle'
    and e.Department = 'Telesales'
```

ORACLE®

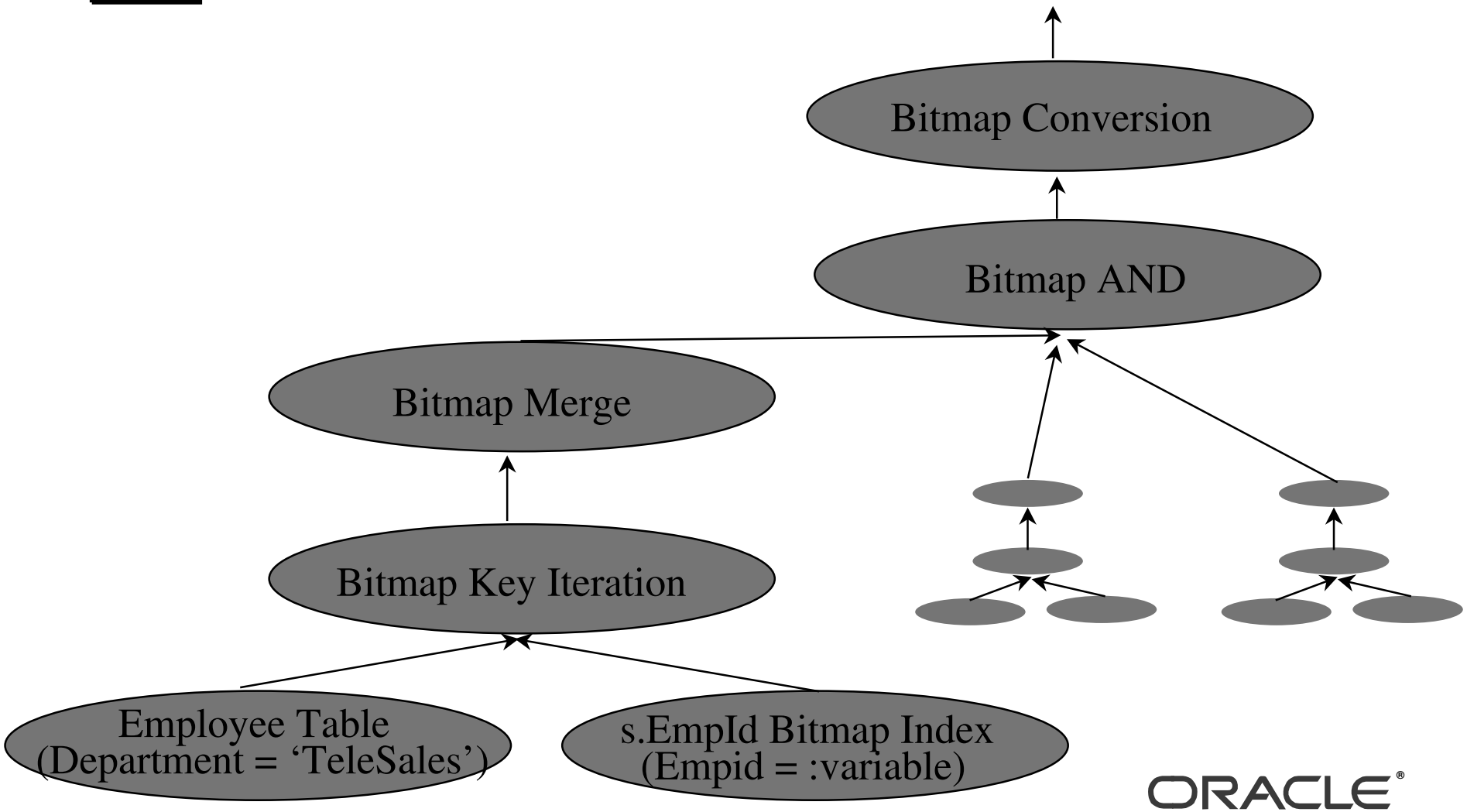


Star Join Queries (cont'd)

```
select c.Name, s.Price
from Employee e, Product p, Customer c, Sales s
where e.EmpId = s.EmpId
    and p.ProductId = s.ProductId
    and c.CustomerId = s.CustomerId
    and c.Income > 100000
    and p.Supplier = 'Oracle'
    and e.Department = 'Telesales'
    and s.ProductcId in (select p.ProductcId
                         from Product p
                         where p.Supplier = 'Oracle')
    and s.EmpId in (select e.EmpId from Employee e
                    where e.Department = 'Telesales')
    and s.CustomerId in (select c.CustomerId from Customer c
                         where c.Income > 100000)
```

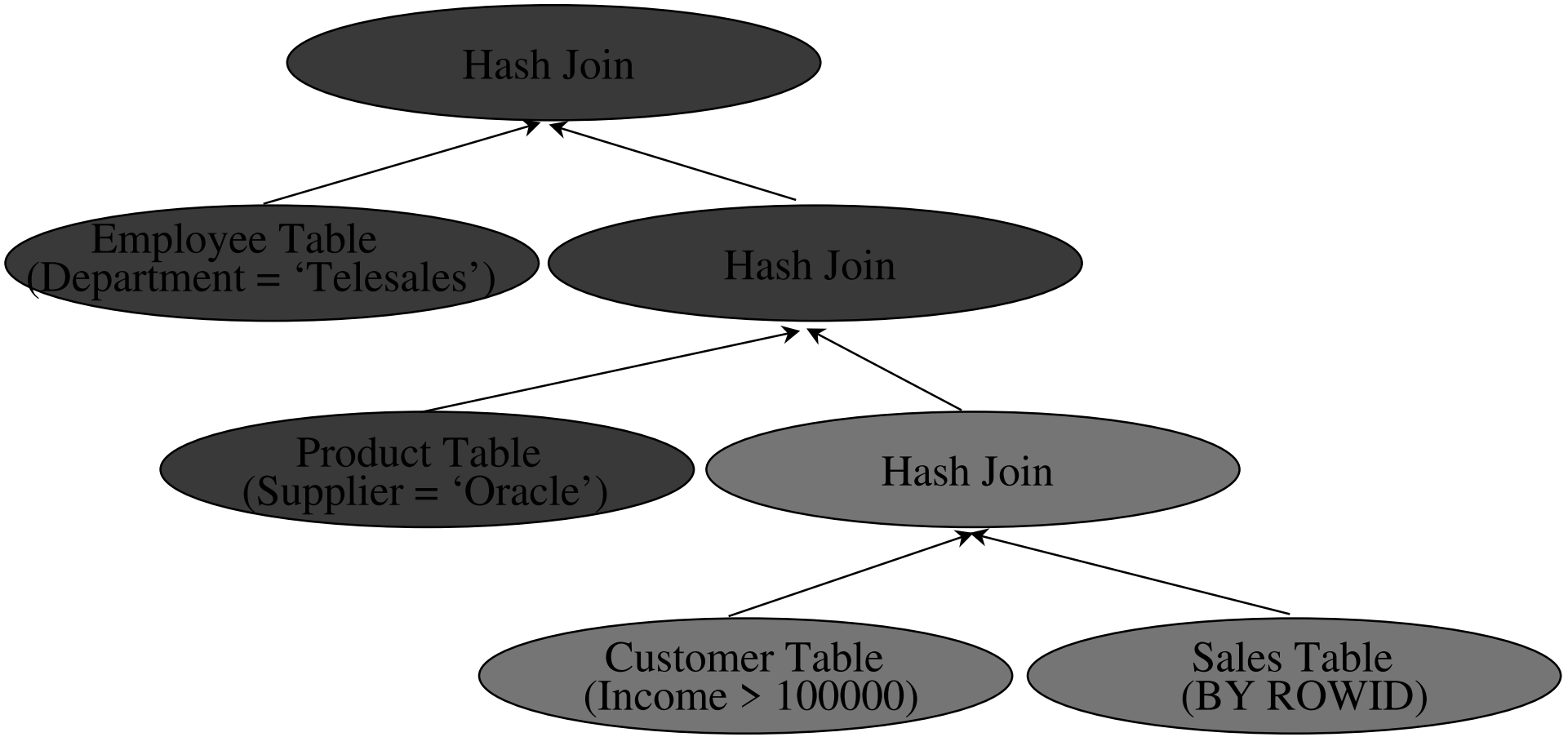
ORACLE®

Star Join Queries (cont'd)

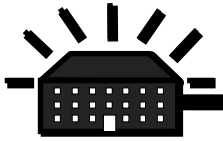


ORACLE®

Star Join Queries (cont'd)



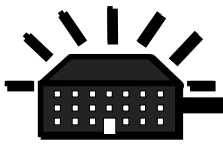
ORACLE®



Star Join Queries (cont'd)

- **Flexibility: One single-column index per dimension key on fact table.**
- **Performance: Can handle many dimensions, sparsity, etc**
- **Parallel scalability. Parallelized by fact table rowid ranges.**
- **Supports multiple fact tables, snowflake schemas, views, remote dimension tables, etc.**
- **Optimizer decides what dimensions to use. May choose a completely different access path if better.**

ORACLE®



Conclusion

- Bitmap indexing essential component in RDBMS for data warehousing.
- Main advantages: Space utilization and processing of complex queries.
- Can be used for star query processing.
- Should support updates of individual rows.
- Many advantages to have an integrated solution that can do both OLTP and data warehousing.

ORACLE®